DHBW
Duale Hochschule
Baden-Württemberg

Grundlagen KI – Programmentwurf

DOKUMENTATION

Matrikelnummer, Kurs	5697407, INF21C
Betreuer	Prof. Dr. Dirk Reichardt
Einzusetzende Methode	Evolutionary Computing (EV1)

INHALTSVERZEICHNIS

1 GRUNDSÄTZLICHE PROGRAMMSTRUKTUR

Als Programmiersprache für die Umsetzung wurde Java ausgewählt. Diese Sprache bietet sich an, da sie objektorientiert ist und sich aus der Aufgabenstellung relativ intuitiv die verschiedenen Klassen für das Programm ableiten lassen. Diese sind:

- Lkw
- Auftrag
- Individual
- Population
- Beladungsstrategie (enthält main-Methode mit Algorithmus)

Abbildung 1 zeigt das genaue Klassendiagramm mit allen Attributen, Methoden und Beziehungen zwischen den Klassen (ausgenommen Getter- und Setter-Methoden).

UML Diagramm

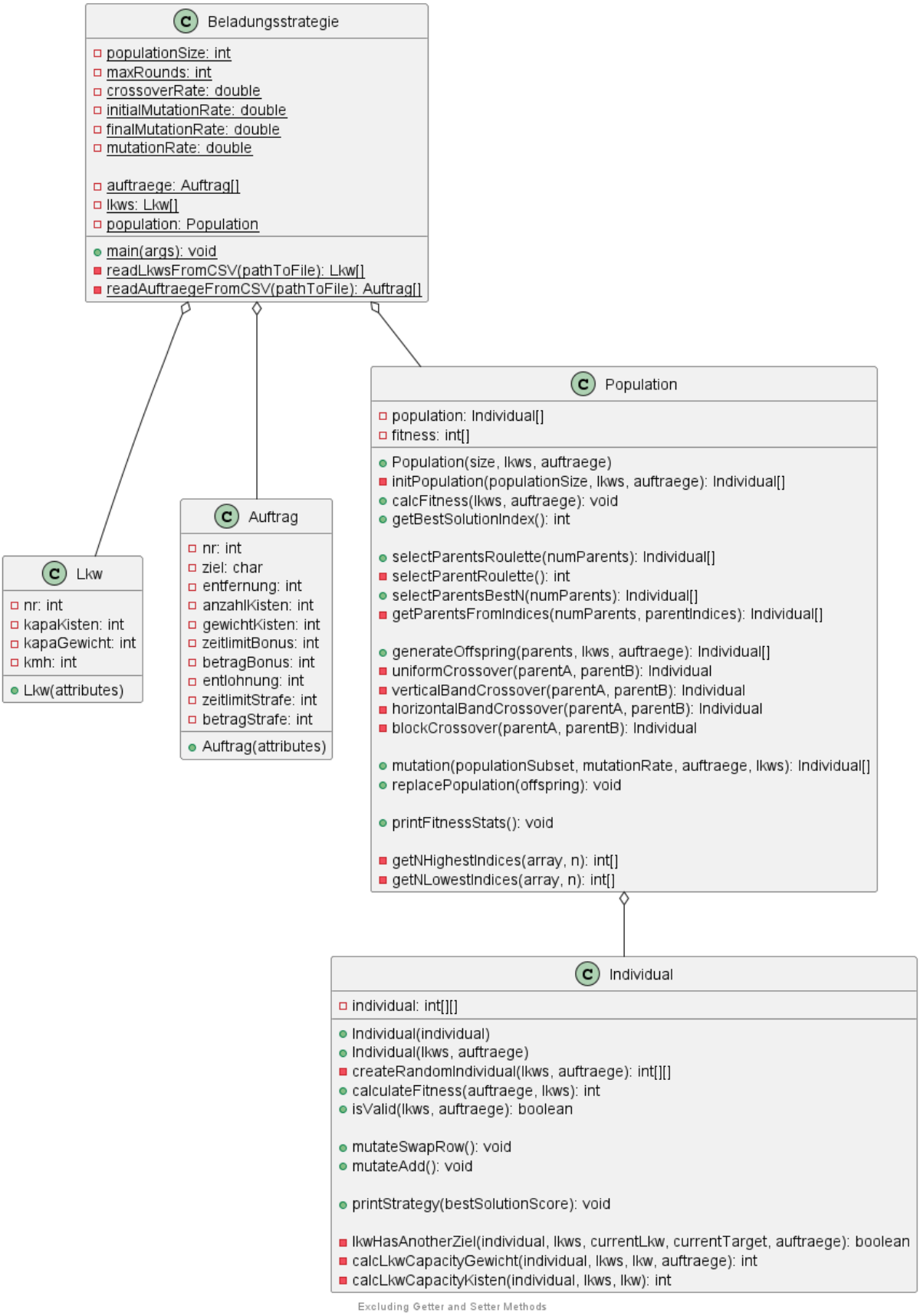

Abbildung 1: Klassendiagramm des Projekts

2 UMSETZUNG

2.1 EINLESEN DER AUFTRÄGE UND LKWS

Mithilfe eines `BufferedReaders` werden die beiden Dateien „lkw_2.csv“ und „auftraege_2.csv“ ausgelesen. Jede Zeile wird dabei in ein Objekt der Klasse Lkw bzw. Auftrag umgewandelt. Die Objekte werden anschließend in jeweils einem Array pro Klasse gespeichert, sodass im späteren Programmverlauf darauf zugegriffen werden kann.

2.2 REPRÄSENTATION DER INDIVIDUEN

Um mögliche Beladungsstrategien (also die Individuen des Algorithmus) zu repräsentieren, werden 2-dimensionale Integer Arrays verwendet. Dabei steht jede Zeile für einen Lkw und jede Spalte für einen Auftrag. Ein Eintrag an der Stelle `individuum[i][j] = n` bedeutet dann, dass n Kisten vom Lkw i für den Auftrag j befördert werden. Dabei dürfen Aufträge von mehr als einem Lkw ausgefahren werden und Lkw darf Kisten für mehrere Aufträge ausfahren (solange diese das gleiche Ziel haben). Ein Beispiel für eine solche Repräsentation ist sichtbar in Tabelle 1.

Lkw \ Auftrag	0	1	2	3	4	5	6
0	0	0	1	0	0	2	0
1	0	5	0	0	0	3	0
2	0	3	0	0	7	0	0
3	0	0	5	0	0	0	10

Tabelle 1: Beispiel für Repräsentation einer Beladungsstrategie

2.3 INITIALISIERUNG DER ANFANGSPOPULATION

Die Schwierigkeit bei der Generierung der Anfangspopulation besteht darin, dass die Lösungen den Rahmenbedingungen der Aufgabenstellung folgen müssen, also nur Lösungen generiert werden dürfen, die auch eine erlaubte Lösung des Problems darstellen. Diese Bedingungen sind:

1. Ein Lkw darf nicht mehr Kisten fahren als erlaubt (Kapa_Kisten)
2. Ein Lkw darf nicht mehr Gewicht fahren als erlaubt (Kapa_Zuladung)
3. Falls mehrere Aufträge auf einem Lkw transportiert werden, müssen diese das gleiche Ziel haben

Für die Initialisierung der Anfangspopulation wurden zwei Optionen ausprobiert.

2.3.1 Zufällige Generierung

Generierung von zufälligen Individuen, anschließende Überprüfung ob alle Rahmenbedingungen erfüllt werden. Dies wird solange wiederholt bis die gewünschte Populationsgröße erreicht ist.

Dieser Ansatz stellte sich als nicht zielführend heraus, da die zufälligen Lösungen so gut wie nie die Rahmenbedingungen erfüllen, sodass alleine die Initialisierung extrem ineffizient und zeitintensiv wäre.

2.3.2 Randomisierte Generierung nach Regeln

Bei der Generierung der Lösungen werden die Kapazitätsgrenzen der Lkws als auch die Restriktionen der Auftragsziele beachtet. Dabei werden nacheinander zufällig Aufträge ausgewählt, die befüllt werden sollen. Für jeden Auftrag wird ein zufälliger Lkw gewählt, der so viele Kisten des Auftrags ausfährt, bis der Auftrag entweder erfüllt ist, oder die Kapazitätsgrenze des Lkws erreicht ist. Im zweiten Fall wird so lange auf zufälliger Basis ein nächster Lkw ausgewählt, der noch kein anderes Ziel anfährt, bis alle Kisten des Auftrags vergeben sind, oder kein Lkw mehr für diesen Auftrag in Frage kommt. Auf diese Weise können erfolgreich und zeiteffizient zufällige Lösungen generiert werden, die bereits sinnvoll sind und die Restriktionen des Problems einhalten.

2.4 Bewertung der Individuen (Fitnessfunktion)

Für die Bewertung der Fitness der Lösungen gibt es mehrere Möglichkeiten nach denen optimiert werden kann, z.B. die Zeit, die vergeht bis alle Lkws zurück sind oder die Anzahl der Kisten, die transportiert werden. Für ein Unternehmen ist im betriebswirtschaftlichen Sinne letztendlich die Gewinnfunktion ausschlaggebend, das heißt wie viel Geld am Ende erwirtschaftet wird, weshalb diese auch in diesem Kontext verwendet wird. Die Gewinnfunktion wird berechnet aus den Werten für Strafe, Entlohnung und Bonus unter Berücksichtigung der verstrichenen Zeit, die die Lkws für die Erfüllung eines Auftrags benötigen. Je höher der Gewinn, desto besser ist auch die generierte Lösung.

2.5 Selektion der Eltern zur Fortpflanzung

Für die Selektion wird das Roulette Prinzip verwendet. Das heißt, dass Individuen mit einem höheren Fitness Wert eine proportional höhere Chance haben als Eltern ausgewählt zu werden. Um die Varianz innerhalb der Population jedoch beizubehalten, können auch Individuen mit einer niedrigeren Fitness vorkommen, allerdings nur mit einer geringeren Wahrscheinlichkeit.

Die Implementierung dieses Prinzips funktioniert so:

1. Aufaddieren aller Fitnesswerte der Population (`totalFitness`)
2. Generiere Zufallszahl zwischen `0` und `totalFitness`
3. Mappe die Zufallszahl mit dem Individuum, in dessen Range die Zufallszahl liegt
4. Das ermittelte Individuum wird zur Fortpflanzung verwendet
5. Wiederhole Schritte 2-4 solange bis genug Eltern ermittelt wurden

2.6 REKOMBINATION

Aus den n ermittelten Eltern werden anschließend n neue Individuen generiert. Dafür wurden mehrere Crossover Algorithmen ausprobiert, welche auf den Crossover Operatoren in https://content.wolfram.com/sites/13/2018/02/05-3-4.pdf aufbauen.

Individuen, die durch Crossover entstehen werden zuerst geprüft, ob sie auch die Rahmenbedingungen für valide Repräsentation erfüllen. Nur dann werden sie in die Population aufgenommen. Die implementierten Crossover Operatoren werden im Folgenden beschrieben.

2.6.1 Horizontal Band Crossover

Ein zufällig gewähltes horizontales Band der Repräsentation von Parent A wird in Parent B eingefügt.

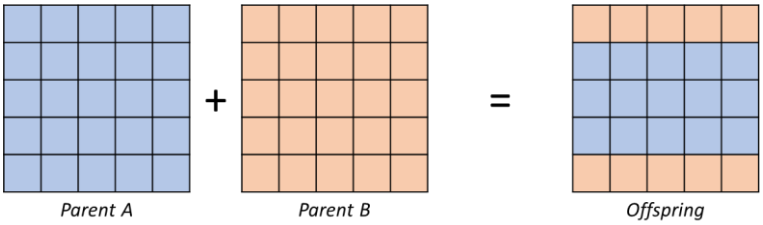

Abbildung 2: Horizontal Band Crossover

Bei Ablauf des Algorithmus zeigte dieser Crossover Operator die besten Ergebnisse und wurde deshalb auch für die finale Version verwendet. Das liegt wahrscheinlich daran, dass bei dieser Option die Aufteilung auf die einzelnen Aufträge innerhalb der Lkws nicht aufgebrochen wird und die Kapazitäts- und Zielgrenzen dadurch eher eingehalten werden.

2.6.2 Vertical Band Crossover

Ein zufällig gewähltes vertikales Band der Repräsentation von Parent A wird in Parent B eingefügt.

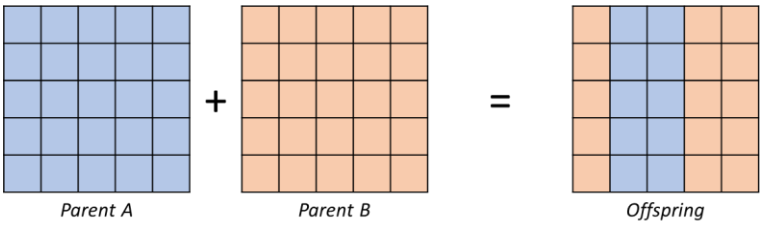

Abbildung 3: Vertical Band Crossover

2.6.3 Block Crossover

Ein zufällig gewählter zusammenhängender Block beliebiger Größer der Repräsentation von Parent A wird in Parent B eingefügt.

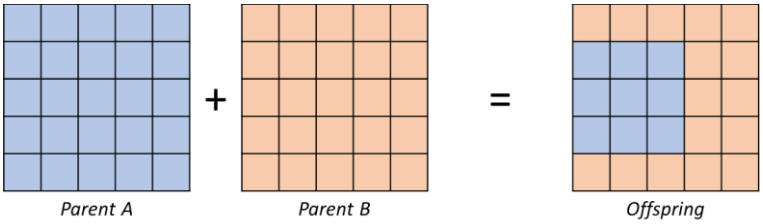

Abbildung 4: Block Crossover

2.6.4 Uniform Crossover

Jede Stelle des 2D-Arrays wird zu gleicher Wahrscheinlichkeit aus Parent A oder Parent B übernommen.

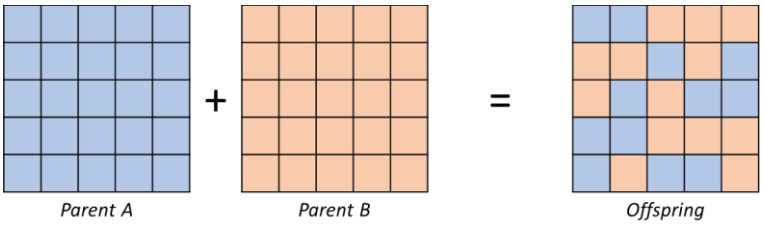

Abbildung 5: Uniform Crossover

2.7 MUTATION

Bevor die neu generierten Individuen in die Population eingefügt werden, wird ein gewisser zufällig ausgewählter Anteil davon mutiert, also leicht abgeändert, um mehr Variation beizubehalten. Die Mutationsrate hat dabei einen relativ niedrigen Startwert und wird in jeder Runde leicht erhöht, sodass immer mehr Individuen mutieren. So werden am Anfang, wenn die Varianz innerhalb der Population ohnehin schon hoch ist, weniger neue Individuen mutiert, sodass sich das System darauf konzentrieren kann, die besten Lösungen aus den vorhandenen Ideen zu extrahieren. Später, wenn die Lösungen innerhalb der Population sich ohnehin schon mehr ändern und eine höhere Qualität aufweisen, soll durch die höhere

Mutationsrate versucht werden, durch viele kleinere Variationen noch mehr aus den Lösungen herauszuholen.

Die Mutation eines Individuums erfolgt in zwei hintereinander ausgeführten Schritten:

1. Innerhalb einer beliebigen Spalte werden zwei Einträge ausgetauscht.
2. An einer beliebigen Stelle wird der dortige Wert um 1 erhöht.

Die Mutation wird nur dann durchgeführt, wenn das Ergebnis wiederum alle Rahmenbedingungen für Individuen erfüllt.

2.8 Ersetzung

Die `n` schlechtesten Individuen werden durch die `n` neu generierten Lösungen ersetzt, sodass die Populationsgröße während des gesamten Ablaufs gleichbleibt.

2.9 Terminierungskriterium

Der Algorithmus läuft in Runden ab. Nach dem Einlesen der Lkws und Aufträge und der Initialisierung der Population (nach der Methode der randomisierten Generierung nach Regeln) läuft jede Runde wie folgt ab:

1. Bewertung der Fitness
2. (Falls neue beste Lösung generiert wurde: Merke diese Lösung)
3. Selektion der Eltern
4. Rekombination (Mit horizontal crossover operator)
5. Mutation
6. Ersetzung

Nach einer vorher festgelegten Obergrenze `maxRounds` terminiert der Algorithmus. Die zu diesem Zeitpunkt als beste Lösung gespeicherte Repräsentation wird ausgegeben und ist die beste Beladungsstrategie, die vom Algorithmus ermittelt werden konnte.

2.10 Hyperparameter

Die empirisch ermittelten besten Hyperparameter für den Algorithmus sind:

```
populationSize = 5000
maxRounds = 500
crossoverRate = 0.2
initialMutationRate = 0.2
finalMutationRate = 0.6
```

3 AUSWERTUNG DER ERGEBNISSE

Die beste Lösung die im Laufe der Versuche durch den Algorithmus erreicht werden konnte, besitzt einen Fitness-Wert von 47095 und ist abrufbar im Dokument „Auswertung.xlsx“.

Aus den Auftragsdaten wurde ermittelt, dass der maximal erreichbare Profit bei insgesamt 89670, bzw. ohne Boni bei 77770 liegt. Die ermittelte Lösung erreicht also 53% bzw. 61% dieses maximalen Werts.

Betrachtet man die limitierenden Faktoren der Aufgabenstellung, nämlich die Kistenkapazität und Gewichtskapazität, wird deutlich, dass die Aufträge zwar vorsehen, dass 535 Kisten transportiert werden, die Lkws aber insgesamt nur Kapazität für 258 aufweisen. Das sind nur 48% der eigentlich erforderten Kapazität. Zudem können die Lkws nur 72% des zu transportierenden Gewichts transportieren. Die Tatsache, dass der Algorithmus 53% bzw. 61% des maximalen Profits erreicht hat, obwohl nach Anzahl der Kisten nur ca. 48% erreichbar sein sollten, zeigt, dass die Lösung durchaus sinnvoll ist.

Zusätzlich zeigt die genauere Betrachtung der Lkw-Beladungen, dass die Lkws alle voll beladen sind (entweder in Bezug auf Kisten oder Gewicht), was für die Effizienz der Lösungen spricht. Aus diesen Gründen ist zu vermuten, dass der Algorithmus tatsächlich das Beste aus den gegebenen Rahmenbedingungen herausgeholt hat, und die ermittelte Lösung qualitativ gut ist.